

LTSI: Long Term Stable Kernel and it's Testing

Tsugikazu Shibata & Hisao Munakata

Linux Foundation Consumer Electronics working group

October 24th 2013

Who am I ?

- From embedded SoC provider company Renesas
- Linux Foundation CE¹ working Gr. Steering committee member, LF/CEWG Architecture Gr. co-chair
- One of LF/CEWG LTSI² project initial proposer
- At my company, I had been encouraging my team developers to send a patches upstream
- Also I have supported various CE customers who develop digital-TV, Blu-ray recorder and Smart-phone

¹CE = consumer electronics

²LTSI =Long Term Support Initiative

Why you should consider adopting LTS ?

Why you should consider adopting LTS ?
The way to add your code to longterm.
Shared kernel test for LTSI validation
Conclusion

kernel release mechanism
Messages from Greg KH
Whoops, Is it too late ?



Upstream kernel @kernel.org

Protocol	Location
HTTP	https://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:



3.11.6

mainline:	3.12-rc6	2013-10-19	[tar.xz]	[pgp]	[patch]	[view patch]	[cglt]
stable:	3.11.6	2013-10-18	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
stable:	3.9.11 [EOL]	2013-07-21	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	3.10.17	2013-10-18	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	3.4.66	2013-10-13	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	3.2.51	2013-09-10	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	3.0.100	2013-10-13	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	2.6.34.14	2013-01-16	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
longterm:	2.6.32.61	2013-06-10	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc] [cglt] [changelog]
linux-next:	next-20130927	2013-09-27					[cglt]

You can find 1)latest released, 2)under development (=mainline, next), and **several stable kernels**

The release record of 3.0 series upstream kernel

version	release date	duration
v3.0	2011-07-21	
v3.1	2011-10-24	95 days
v3.2	2012-01-04	72 days
v3.3	2012-03-18	74 days
v3.4	2012-05-20	63 days
v3.5	2012-07-21	62 days
v3.6	2012-09-30	71 days
v3.7	2012-12-10	71 days
v3.8	2012-02-18	70 days
v3.9	2013-04-28	69 days
v3.10	2013-06-30	63 days
v3.11	2013-09-02	64 days

Release happened regularly at around every 70 days

However, not all kernels are maintained for longterm

version	maintenance status
v3.0	longterm (3.100) -> EOL soon
v3.1	maintained till 3.1.9, then now EOL
v3.2	longterm (3.2.51), kept maintained
v3.3	maintained till 3.3.8, then now EOL
v3.4	longterm (3.4.66), kept maintained
v3.5	maintained till 3.5.7, then now EOL
v3.6	maintained till 3.6.11, then now EOL
v3.7	maintained till 3.7.10, then now EOL
v3.8	maintained till 3.8.13, then now EOL
v3.9	maintained till 3.9.11, then now EOL
v3.10	stable (3.10.17) -> move to longterm mode soon
v3.11	stable release (3.11.6) till 3.12 release

Stable release include **MUST APPLY** essential fixes

version	fixes
v3.0 -> v3.0.99	3,907
v3.1 -> v3.1.9	647
v3.2 -> v3.2.51	4,055
v3.3 -> v3.3.8	698
v3.4 -> v3.4.66	3,297
v3.5 -> v3.5.7	816
v3.6 -> v3.6.9	676
v3.7 -> v3.7.10	718
v3.8 -> v3.8.13	996
v3.9 -> v3.9.11	746
v3.10 -> v3.10.17	1,141
v3.11 -> v3.11.6	452

Stable kernel rules

(/Documentation/stable_kernel_rules.txt)

Rules on what kind of patches are accepted,
and which ones are not, into the "-stable" tree:

- It must be obviously correct and tested.
- It cannot be bigger than 100 lines, with context.
- It must fix only one thing.
- It must fix a real bug that bothers people (not a, "This could be a problem..." type thing).
- It must fix a problem that causes a build error (but not for things marked CONFIG_BROKEN), an oops, a hang, data corruption, a real security issue, or some "oh, that's not good" issue. In short, something critical.

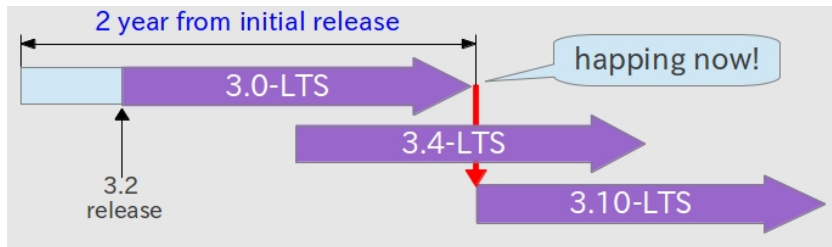
:

- 1) Proven (already merged) code only
- 2) Serious bug fix only
- 3) Serious security fix only

Longterm stable (LTS) kernel release cadence

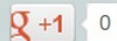
Target kernel selection rules

- Maintainer will **choose one LTS version per year**
- **Maintain it for 2 years** from its original release
- Then, we have 2 LTS kernels like 3.0 and 3.4



LTS/LTSI maintainer, Greg says 3.0 moves to EOL

Date Sun, 13 Oct 2013 15:19:54 -0700
From Greg KH <>
Subject Linux 3.0.100



NOTE! The 3.0.x kernel series will be moving to End-Of-Life soon,
within a week. Please move anything that is relying on this kernel
version to the other longterm kernel releases (3.4.x or 3.10.x) as soon
as possible. If anyone has any questions about this, please let me
know.

I'm announcing the release of the 3.0.100 kernel.

All users of the 3.0 kernel series must upgrade.

<https://lkml.org/lkml/2013/10/13/160>

And, Greg also announced next longterm is **3.10**

Longterm kernel 3.10



By Greg KH - August 4, 2013 - 4:45am

As I've [discussed in the past](#), I will be selecting one "longterm stable" kernel release every year, and maintain that kernel release for at least two years.

Despite the fact that the 3.10-stable kernel releases are not slowing down at all, and there are plenty of pending patches already lined up for the next few releases, I figured it was a good time to let everyone know now that I'm picking the 3.10 kernel release as the next longterm kernel, so they can start planning things around it if needed.

<http://www.linuxfoundation.org/news-media/blogs/browse/2013/08/longterm-kernel-310>

Upstream kernel 3.10 development (is done)

Whoops, we can not submit our latest device support code to 3.0 kernel now! Yes, that is true, because

item	date
kernel 3.10 merge window open	2013.4.28
kernel 3.10 merge window close	2013.5.12
kernel 3.10 release	2013.6.30

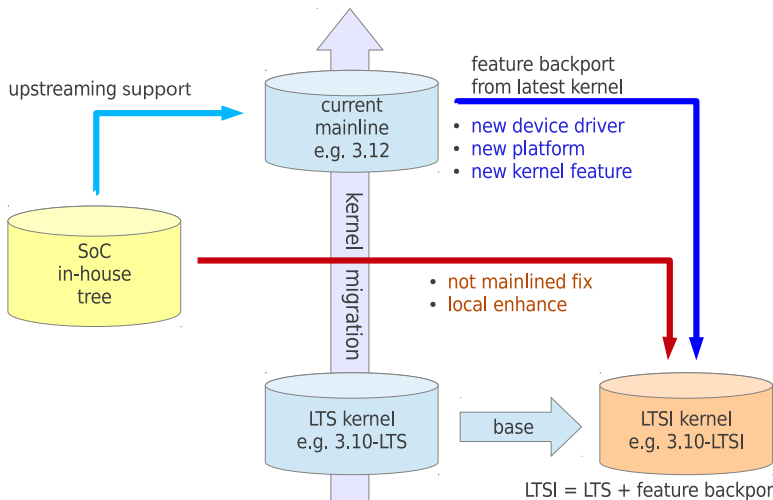
As upstream 3.10 patch merge window is already closed, there is no chance to add your code to upstream kernel. Thus a cutting-edge silicon release after development cycle can not be supported in longterm 3.10 kernel. This might be problematic for embedded industry Linux adopter.

The way to add your code to longterm.


Do not panic, we have prepared another story **LTSI**

item	date
kernel 3.10 merge window open	2013.4.28
kernel 3.10 merge window close	2013.5.12
kernel 3.10 release	2013.6.30
Announce of 2013 LTS kernel version	2013.8.4
LTSI-3.10 git tree open	2013.9.11
3.10 becomes LTS	3.12 release date
LTSI-3.10 merge window open	the same day above
LTSI-3.10-rc1 (=merge window close)	roughly end of 2013
LTSI-3.10-rc2 (=final validation release)	roughly mid Jan 2014
LTSI-3.10 release target	End Jan to mid Feb

LTSI = community LTS(longterm) + industry extra



Yocto and LTSI project coordination is working now




- ABOUT
- ECOSYSTEM
- DOWNLOADS
- TOOLS + RESOURCES
- DOCUMENTATION

Ecosystem
Product Showcase
Yocto Project Participants
Member Organizations
Supporting Organizations
Compliance Program
Compliance Program Registrar

SEARCH [Go](#)

Long Term Support Initiative (LTSI)



LTSI is an industry-wide project created and supported by Hitachi, LG Electronics, NEC, Panasonic, Qualcomm Atheros, Renesas Electronics Corporation, Samsung Electronics, Sony and Toshiba and hosted at The Linux Foundation to maintain a common Linux base for use in a variety of consumer electronics products. The project creates and maintains a long-term industry tree, which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2-3 years.

This new initiative is crucial because device makers are doing significant back-porting, bug testing and driver development on their own, which carries substantial cost in terms of time-to-market, as well as development and engineering effort to maintain those custom kernels. Through collaboration in this initiative, these CE vendors will reduce the duplication of effort currently prevalent in the consumer electronics industry.

The LTSI tree is expected to be a usable base for the majority of embedded systems, as well as the base for ecosystem players (e.g., semiconductor vendors, set-vendors, software component vendors, distributors, and system/application framework providers). The LTSI project will combine the innovative features in newer kernels needed by CE vendors with a stable kernel, while helping those vendors get their code upstream to benefit the entire Linux community. The goal is to reduce the number of private trees currently in use in the CE industry and encourage more collaboration and sharing of development resources.

Discipline of LTSI project management

- Community **LTS + industry demanded** extra patches.
- **Governed by LF/CEWG**
- **Focus on kernel** code^a, not aiming complete BSP
- Therefore, can be combined with existing platform^b
- **CPU architecture and platform neutral**
- **Comply with upstream** rules^c
- Industry friendly acceptance (**flexible patch forms**, etc)
- Help CE (and others) industry to utilize Linux

^adevice drivers are part of kernel, of course

^bAndroid, **Yocto**, Tizen, AGL, WebOS and others

^ce.g. signed-off-by process

LTSI value proposition (1) For CE Industry (part 1)

- **Efficient and less-expensive operations** in enabling Linux to a variety of devices with **Less fragmented use of Linux**
 - Easy to enable Linux on a variety of devices with less resources.
 - Easy to apply latest bug fixes, and determine problems.
 - As **the differences between the LTSI tree and the latest version of upstream tree are expected to be small** due to feature back-porting, industry engineers can work with community engineers to test and fix on the latest version easily.
- **No need to apply some local fixes anymore**
 - Some local fixes will be either incorporated into upstream through the industry staging tree and back-ported into the LTSI tree by the LTSI BP team, or, directly incorporated into the LTSI tree and upstreamed by an LTSI upstreaming engineer.

LTSI value proposition (1) For CE Industry (part 2)

- **Additional features** which are beneficial for the industry are available in the **LTSI** tree.
 - The LTSI project accepts features which are **not incorporated into upstream yet but are very beneficial for the industry**, and continuously works with an author to upstream it. Therefore, the industry can use optimized Linux for the industry.
- More **close interactions with upstream** engineers
 - Through this project, the LTSI team encourages industry engineers to work with upstream engineers, and connects both.

LTSI value proposition (2) For Semiconductor

- Enhancements and bug fixes in upstream will be propagated to the LTSI tree, then, applied to distributions and actual embedded systems. This is a very effective way to enhance and maintain their code.
- Semiconductor vendors can directly contribute to the LTSI tree to integrate latest features, and creates an LTSI based BSP which is ready to be merged into the current product software stack by the set vendors.
- No need to work on an individual tree inside customers privately to merge with the BSP for newly designed SoC.
- Semiconductor vendors can directly contribute to the LTSI tree to integrate latest features, and creates an LTSI based BSP which is ready to be merged into the current product software stack by the set vendors.

LTSI value proposition (3) For distro & community

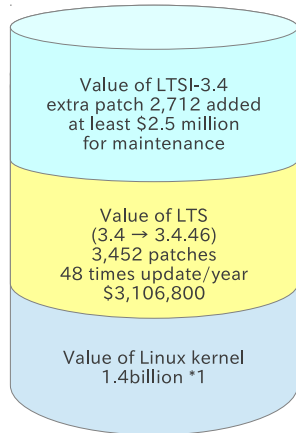
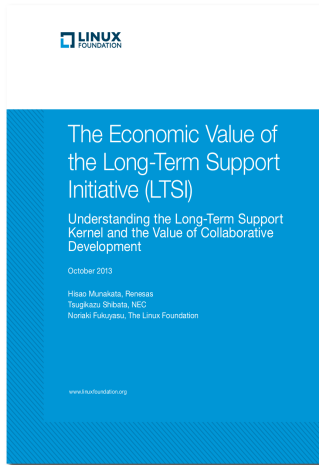
For Distributors

- As the LTSI Back Port Team maintains the LTSI tree up-to-date, distributors **can focus on their differentiations.**

For the Linux Community

- Better communications with industry engineers with a common code base.
- **More contributions are expected** from the consumer electronics industry.

Whitepaper The economic Value of the LTSI



<http://www.linuxfoundation.org/publications/workgroup/value-of-ltsi>

Shared kernel test for LTSI validation

Design of kernel test (is not trivial)

■ Purpose

- Functional test (as a release criteria)
- Performance test (for check trends, finding regression)
- Stress test (for finding hidden corner case bug)
- Compatibility test

■ Approach

- Small start (bottom up), community style
- Top down (detail & comprehensive), industry style

■ Method

- Hardware independent feature test
(software test with complete hardware abstraction)
- H/W (CPU architecture, chip and board) specific test
- Automated test, or on-demand manual test

We found well written Linux Kernel Tester's Guide

Linux Kernel Tester's Guide (version 0.3)

Michal Piotrowski

michal.k.k.piotrowski@gmail.com

Coauthors:

Maciej Rutecki

maciej.rutecki@gmail.com Unixy.pl

Rafael J. Wysocki

rjw@sisk.pl

English translation by

Rafael J. Wysocki

rjw@sisk.pl

Archive found at

http://www.kerneltravel.net/downloads/tester_guide.pdf

Contents of ``Linux Kernel Tester's Guide'' (1/3)

- 1 The kernel, patches, trees and compilation
 - 1.1 The kernel
 - 1.2 Patches
 - 1.3 Ketchup
 - 1.4 Trees
 - 1.5 The -mm tree
 - 1.6 Compilation and installation
 - 1.6.1 Kernel compilation
 - 1.6.2 Useful make options
 - 1.6.3 Kernel modules
 - 1.6.4 Kernel hacking options
 - 1.6.5 Magic SysRq
 - 1.6.6 Installation
 - 1.6.7 Automated configuration and installation

Contents of ``Linux Kernel Tester's Guide'' (2/3)

2 Testing

2.1 Phase One

2.2 Phase Two (AutoTest)

2.3 Phase Three

2.4 Measuring performance

2.5 Hello world! or What exactly are we looking

2.6 Binary drivers and distribution kernels

3 Collecting kernel messages

3.1 Syslog, console and dmesg

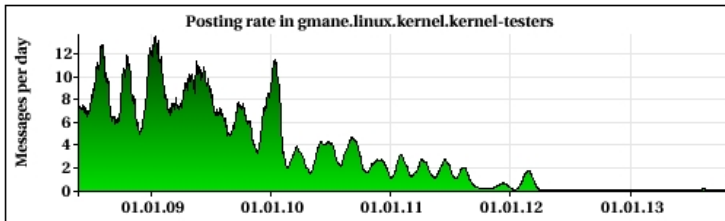
3.2 Serial console

3.3 Network console

Contents of ``Linux Kernel Tester's Guide'' (3/3)

- 4 Git
 - 4.1 Git
 - 4.2 Quilt
 - 4.3 General idea of binary searching
 - 4.4 Binary searching with the help of quilt
 - 4.5 Binary searching with the help of git-bisect
 - 4.6 Caveats
- 5 Reporting bugs
- 6 Testing of hardware
- A Related topics
 - A.1 Test system
 - A.2 KLive
 - A.3 Sending patches
 - A.4 How to become the kernel developer?
- B License

However, kernel-testers ML seems getting inactive



ML (kernel-testers) archives =
<http://www.spinics.net/lists/kernel-testers/>

Enterprise kernel validation and hardening are surely taken cared by distro, and they can use distro kernel for the server products. However, there is **no such major distro who can perform kernel validation for embedded**. We need to establish test mechanism.

Who should, and who can perform kernel test?

- Current **enterprise industry** situation
 - Major distribution do a lot of test before release.
 - Enterprise user can rely on distro's validation and support.
- Current **embedded industry** situation
 - No (more) major embedded Linux distro who validate kernel. (yocto is not a distribution, but powerful tools)
 - Product producers spend huge effort for product testing.
 - But **test experiences are not shared, lots of duplication.**
 - Open source developers can not test as they do not have access to real environment.
- Test case may possibly come from industry

Linux adaptor must test kernel with their use case, as each kernel configurations are not identical.

Open Source is not just a source code

- Be able to **share experiences, bugs, knowledge**
- Help each other **under the Open Source License**
- That may make great cost reduction
- During two years of discussion among the industry, lot of industry people asked us to how to increase/keep quality of Linux

Let's start **discussion for kernel test case & status sharing** on top of LTSI collaboration. This can be applied for LTSI release validation, but not limited to. Because **kernel testing must be a common interest across the community and industry.**

On-going trial (1) : test target, method

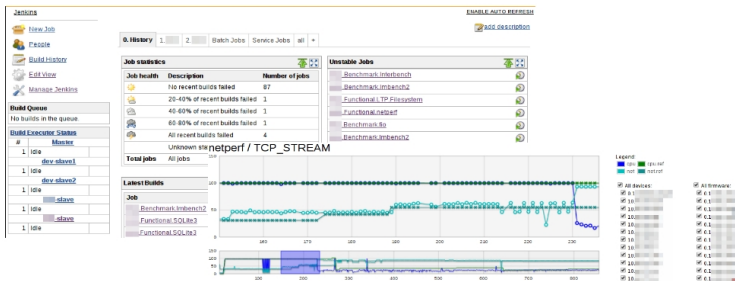
- Functionality
 - Selected LTP tests (e.g. Devices, Filesystems, POSIX)
 - LTP-DDT (see <http://processors.wiki.ti.com/index.php/LTP-DDT>)
 - Netperf (and similar)
- Benchmarks
 - Over 50-70 benchmarks
 - Filesystems, Graphics, Networking
 - Complex benchmarks (e.g. Imbench)
- Stability/Robustness
 - Power failures, stress loads
 - Bad events injection (out of memory, disk wear, etc.)
 - Security
- Vulnerabilities
 - Third-party/Commercial tool/test suite

On-going trial (2) : current setup

- Hardware
 - Renesas R-Car gen1 (BOCK-W or MARZEN)
 - Renesas R-Car gen2 (Lager or Koelsch) in progress
- Software
 - LTSI 3.4, (& 3.10 in near future)
 - Add-on patches (platform specific)
- Setup
 - Jenkins
 - Cogent framework + pre-integrated tests + LTP-DDT
 - Remote access to the test server, power switch, board debug serial, ethernet
- Access can be provided for selected users to evaluate both frameworks.

On-going trial (3) : how it looks like

- **Thin** back-end (target setup/tests integration) + Jenkins (UI, management, configuration)
- Tried to make it **as simple** as running bash script
- More than 50 test suites integrated (**fully automated**)
- Friendly UI, Graphs, Thresholds



Conclusion

Conclusion

- **Correct understanding of the longterm (LTS) cadence** is important. LF/CEWG develops LTSI version on top of community longterm kernel. You can gain huge cost reduction if you can fully utilize LTS & LTSI scheme.
- 3.0 longterm maintenance cycle is a move to EOL soon, and **3.10 is the next longterm support target**. ALSO, **LTSI-3.10 patch collection will start soon** to create LTSI-3.10 release.
- We will setup new kernel validation environment so that industry people can share test case and result. As a next step of industry Linux adapter's collaboration, we want to **create new eco-system for kernel testing**.

Resources

- project web = ltsi.linuxfoundation.org
- ML
 - ML subscription =
<https://lists.linuxfoundation.org/mailman/listinfo/ltsi-dev>
 - ML archives =
<http://lists.linuxfoundation.org/pipermail/ltsi-dev/>
 - ML patchwork =
<https://patchwork.kernel.org/project/ltsi-dev/list/>
- git(each patch) = <http://git.linuxfoundation.org/?p=ltsi-kernel.git;a=summary>
- download (tar ball) =
<http://ltsi.linuxfoundation.org/downloads/releases>
- twitter = @LinuxLTSI
- document archives =
<http://ltsi.linuxfoundation.org/resources>